day034 会话技术

> 作者: 张大鹏

001.会话

```
1. 会话：一次会话中包含多次请求和响应。
    * 一次会话：浏览器第一次给服务器资源发送请求，会话建立，直到有一方断开为止
2. 功能：在一次会话的范围内的多次请求间，共享数据
3. 方式：
    1. 客户端会话技术：Cookie
    2. 服务器端会话技术：Session
```

002.Cookie

> 概念：客户端会话技术，将数据保存到客户端
>
> 原理: 基于响应头set-cookie和请求头cookie实现

快速入门

```
1. 创建Cookie对象，绑定数据
    * new Cookie(String name, String value)
2. 发送Cookie对象
    * response.addCookie(Cookie cookie)
3. 获取Cookie，拿到数据
    * Cookie[]  request.getCookies()
```

003.Cookie常见问题

1. 一次可不可以发送多个cookie?

```
可以
可以创建多个Cookie对象，使用response调用多次addCookie方法发送cookie即可。
```

2. cookie在浏览器中保存多长时间?

```
1. 默认情况下，当浏览器关闭后，Cookie数据被销毁
2. 持久化存储
    setMaxAge(int seconds)
    正数：将Cookie数据写到硬盘的文件中。持久化存储。并指定cookie存活时间，时间到后，cookie文件自动失效
    负数：默认值
    零：删除cookie信息
```

3. cookie能不能存中文?

```
在tomcat 8 之前 cookie中不能直接存储中文数据。
在tomcat 8 之后，cookie支持中文数据。特殊字符还是不支持，建议使用URL编码存储，URL解码解析
```

4. cookie共享问题?

```
1.假设在一个tomcat服务器中，部署了多个web项目，那么在这些web项目中cookie能不能共享?
    默认情况下cookie不能共享
    setPath(String path):设置cookie的获取范围。默认情况下，设置当前的虚拟目录
    如果要共享，则可以将path设置为"/"

2. 不同的tomcat服务器间cookie共享问题?
    setDomain(String path):如果设置一级域名相同，那么多个服务器之间cookie可以共享
    setDomain(".baidu.com"),那么tieba.baidu.com和news.baidu.com中cookie可以共享
```

5. Cookie的特点和作用

```
1. cookie存储数据在客户端浏览器
2. 浏览器对于单个cookie 的大小有限制(4kb) 以及 对同一个域名下的总cookie数量也有限制(20个)

作用:
    1. 访问一个Servlet，如果是第一次访问，则提示：您好，欢迎您首次访问。
    2. 在不登录的情况下，完成服务器对客户端的身份识别
```

004.Cookie的入门使用

1.设置Cookie

```java
package com.lxgzhw.web.cookie;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/*
设置cookie
 */
```

```
@WebServlet("/demo01")
public class Demo01 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {

    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req,resp);
    }
}
```

2.访问`Cookie`

```
package com.lxgzhw.web.cookie;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/*
设置cookie
 */
@WebServlet("/demo02")
public class Demo02 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        System.out.println("正在访问demo02");
        //获取cookie
        Cookie[] cookies = req.getCookies();
        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("msg")) {
                    System.out.println("找到cookie了");
                    System.out.println("cookie=" + cookie.getValue());
                }
            }
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req, resp);
    }
```

```
}
```

005.设置Cookie

1.设置`Cookie`存活1周,且同域名,根目录下可用

```
package com.lxgzhw.web.cookie;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/*
设置cookie
 */
@WebServlet("/demo03")
public class Demo03 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        Cookie time = new Cookie("time", "设置存活时间为7天");
        Cookie where = new Cookie("where", "同顶级域名可用");
        Cookie where1 = new Cookie("where1", "根目录下可用");

        //1.设置存活时间
        time.setMaxAge(3600*24*7);

        //2.设置根目录可用
        where1.setPath("/");

        //3.设置同顶级域名可用
        where.setDomain("lxgzhw.com");

        //4.转发到demo4,从demo4获取cookie
        resp.addCookie(time);
        resp.addCookie(where);
        resp.addCookie(where1);
        req.getRequestDispatcher("/demo04").forward(req,resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req,resp);
    }
```

```
}
```

006.案例:记住上一次访问的时间

需求:
如果第一次访问就添加cookie记录时间,如果不是,就显示上次登录时间

```
package com.lxgzhw.web.cookie;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.text.SimpleDateFormat;

/*
记住用户的上次登录时间
分析:
    1.使用cookie记住lastTime
    2.时间需要格式化
 */
@WebServlet("/demo05")
public class Demo05 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //1.定义一个flag,用来判断是否有lastTime
        boolean flag = false;

        //2.获取cookies
        Cookie[] cookies = req.getCookies();
        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("lastTime")) {
                    //找到了
                    //将时间转化
                    long time = Long.parseLong(cookie.getValue());
                    String formatTime = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss")
                            .format(time);
                    System.out.println("上次登录时间:" + formatTime);

                    //更新时间
                    cookie.setValue(System.currentTimeMillis() + "");
                    resp.addCookie(cookie);
                    flag = true;
                }
```

```
            }
        }

        //3.判断flag
        if (!flag) {
            //没有设置,第一次登录
            System.out.println("第一次登录");
            Cookie lastTime = new Cookie("lastTime", System.currentTimeMillis() + "");
            resp.addCookie(lastTime);
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req, resp);
    }
}
```

007.JSP

```
1.Java Server Pages:  java服务器端页面
2.可以理解为：一个特殊的页面，其中既可以指定定义html标签，又可以定义java代码
3.JSP本质上就是一个Servlet
```

008.JSP脚本

```
1. <%  代码 %>：定义的java代码，在service方法中。service方法中可以定义什么，该脚本中就可以定义什么。
2. <%! 代码 %>：定义的java代码，在jsp转换后的java类的成员位置。
3. <%= 代码 %>：定义的java代码，会输出到页面上。输出语句中可以定义什么，该脚本中就可以定义什么。
```

009.JSP的内置对象

```
1.在jsp页面中不需要获取和创建，可以直接使用的对象
2.jsp一共有9个内置对象。
3.常用三个
    request
    response
    out
4.response.getWriter()和out.write()的区别
    1.在tomcat服务器真正给客户端做出响应之前，会先找response缓冲区数据，再找out缓冲区数据
    2.response.getWriter()数据输出永远在out.write()之前
    3.所以能用out就别用response,否则会打乱输出顺序
```

010.JSP版上次访问时间

```jsp
<%@ page import="java.text.SimpleDateFormat" %><%--
  Created by IntelliJ IDEA.
  User: 18010
  Date: 2019/9/1
  Time: 20:40
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>上次登录时间</title>
</head>
<body>
<%
    //1.定义一个flag,用来判断是否有lastTime
    boolean flag = false;

    //2.获取cookies
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (Cookie cookie : cookies) {
            if (cookie.getName().equals("lastTime")) {
                //找到了
                //将时间转化
                long time = Long.parseLong(cookie.getValue());
                String formatTime = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss")
                        .format(time);
                System.out.println("上次登录时间:" + formatTime);

                //更新时间
                cookie.setValue(System.currentTimeMillis() + "");
                response.addCookie(cookie);
                flag = true;
            }
        }
    }

    //3.判断flag
    if (!flag) {
        //没有设置,第一次登录
        System.out.println("第一次登录");
        Cookie lastTime = new Cookie("lastTime", System.currentTimeMillis() + "");
        response.addCookie(lastTime);
    }
%>
</body>
</html>
```

011.Session

> Session的实现是依赖于Cookie的
>
> 概念：服务器端会话技术，在一次会话的多次请求间共享数据，将数据保存在服务器端的对象中

1.设置 Session

```
package com.lxgzhw.web.session;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/*
设置session
 */
@WebServlet("/sessionDemo01")
public class Demo01 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //1.获取session
        HttpSession session = req.getSession();

        //2.存储数据
        session.setAttribute("msg","我终于会使用session技术了.");

        //3.转发到demo02测试
        req.getRequestDispatcher("/sessionDemo02").forward(req,resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req,resp);
    }
}
```

2.获取 Session

```
package com.lxgzhw.web.session;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/*

 */
@WebServlet("/sessionDemo02")
public class Demo02 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //1.获取session
        HttpSession session = req.getSession();

        //2.获取数据
        Object msg = session.getAttribute("msg");

        //3.打印数据
        System.out.println(msg);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req, resp);
    }
}
```

012.设置Session内存地址

1. 当客户端关闭后，服务器不关闭，两次获取session是否为同一个?
    * 默认情况下。不是。
    * 如果需要相同，则可以创建Cookie,键为JSESSIONID，设置最大存活时间，让cookie持久化保存。

```
Cookie c = new Cookie("JSESSIONID",session.getId());
c.setMaxAge(60*60);
response.addCookie(c);
```

实战案例

```
package com.lxgzhw.web.session;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
```

```java
/*

 */
@WebServlet("/sessionDemo03")
public class Demo03 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //1.获取session
        HttpSession session = req.getSession();
        System.out.println(session);

        //2.设置浏览器关闭后,session也相同
        Cookie jsessionid = new Cookie("JSESSIONID", session.getId());
        jsessionid.setMaxAge(3600 * 24);
        resp.addCookie(jsessionid);

        //3.比较第一次访问和关闭浏览器后访问
        Cookie[] cookies = req.getCookies();
        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("JSESSIONID")) {
                    System.out.println(cookie.getValue());
                }
            }
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req, resp);
    }
}
```

013.Session防数据丢失机制

客户端不关闭，服务器关闭后，两次获取的session是同一个吗?

```
不是同一个，但是要确保数据不丢失。tomcat自动完成以下工作
* session的钝化:
    * 在服务器正常关闭之前，将session对象系列化到硬盘上
* session的活化:
    * 在服务器启动后，将session文件转化为内存中的session对象即可。
```

014.Session销毁时间

1. 服务器关闭
2. session对象调用invalidate()
3. session默认失效时间 30分钟

```
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

015.Session的特点

1. session用于存储一次会话的多次请求的数据，存在服务器端
2. session可以存储任意类型，任意大小的数据

016.Session和Cookie的区别

1. session存储数据在服务器端，Cookie在客户端
2. session没有数据大小限制，Cookie有
3. session数据安全，Cookie相对不安全

017.案例:验证码

login.jsp中输入验证码,使用LoginServlet.java判断验证码是否正确

1.Captcha.java 提供验证码

```
package com.lxgzhw.web.servlet;

import javax.imageio.ImageIO;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.Random;

/*

 */
@WebServlet("/captcha")
public class Captcha extends HttpServlet {
```

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //1.定义验证码图片的宽高
        int width = 100, height = 50;

        //2.创建验证码图片对象
        BufferedImage img =
                new BufferedImage(width, height,
                        BufferedImage.TYPE_INT_RGB);

        //3.填充背景色
        Graphics graphics = img.getGraphics();//画笔
        graphics.setColor(Color.pink);//画笔颜色
        graphics.fillRect(0, 0, width, height);//填充矩形

        //4.画边框
        graphics.setColor(Color.blue);
        graphics.drawRect(0, 0, width - 1, height - 1);

        //5.生成随机字符
        //5.1定义验证码
        StringBuffer captcha = new StringBuffer();
        String str = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghigklmnopqrstuvwxyz0123456789";
        Random random = new Random();
        for (int i = 0; i < 4; i++) {
            //5.1验证码每个字符的索引
            int index = random.nextInt(str.length());
            //5.2获取字符
            char c = str.charAt(index);
            captcha.append(c);
            //5.4写验证码
            graphics.drawString(c + "", width / 5 * i + 10, height / 2);
        }
        //5.2将验证码设置为session
        HttpSession session = req.getSession();
        session.setAttribute("captcha",captcha.toString());

        //6.画干扰线
        graphics.setColor(Color.green);
        for (int i = 0; i < 10; i++) {
            int x1 = random.nextInt(width);
            int x2 = random.nextInt(width);
            int y1 = random.nextInt(height);
            int y2 = random.nextInt(height);
            graphics.drawLine(x1, y1, x2, y2);
        }

        //7.输出图片到页面
        ImageIO.write(img, "jpg", resp.getOutputStream());
    }

    @Override
```

```
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req,resp);
    }
}
```

2. `login.jsp`获取验证码并填写验证码

```
<%--
  编辑器: IntelliJ IDEA.
  作者: 18010
  日期: 2019/9/1
  时间: 21:10
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>登录</title>
</head>
<body>
<form action="/login" method="post">
    <table border="1" width="60%" align="center">
        <tr>
            <td>用户名</td>
            <td>
                <input type="text" name="username">
            </td>
        </tr>
        <tr>
            <td>密码</td>
            <td>
                <input type="text" name="password">
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <img src="/captcha" alt="" id="captcha">
            </td>
        </tr>
        <tr>
            <td>验证码</td>
            <td>
                <input type="text" name="captcha">
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <input type="submit" value="确认">
            </td>
        </tr>
    </table>
</form>
<script>
```

```
    window.onload = function (ev) {
        document.getElementById("captcha").onclick = function (ev1) {
            this.src = "/captcha?time=" + new Date().getTime();
        }
    }
</script>
</body>
</html>
```

3. `LoginServlet.java` 判断验证码是否正确

```
package com.lxgzhw.web.servlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/*

 */
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //处理用户登录
        //主要演示处理验证码
        //1.获取验证码
        String captcha = req.getParameter("captcha");
        HttpSession session = req.getSession();
        String captcha1 = (String) session.getAttribute("captcha");

        //2.比较验证码
        if (captcha.equalsIgnoreCase(captcha1)){
            //3.打印信息
            System.out.println("验证码正确");
        }
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        this.doGet(req,resp);
    }
}
```